

~~SECRET~~ ~~CONFIDENTIAL~~

200 RPT HQ CUF
201 RPT URGENT CHANNEL 0
202 PORE 18514.0
203 PAST
204 LAT L-58R 10002
205 RPT R 15 BLUM SCALING
206 FACTOR
207 LAT 201
208 LAT DARKPAGE 10513
209 BLUM
210 DUSUN 130
211 GOLF 1000
212 PRINT

Fig. 1(b)
BASIC LAYOUT
FOR A/D CONVERTER

```

120: INPUT B
121: PRINT C
130: REM MAKE CURRENT B TO B WITH
    REM MULTIPLY
135: LET PRINT (C*100)
137: LET THING (C-100/100/100)
139: LET UNIT (C-100/100)-(C*100)
140: REM MAKE CURRENT C TO C WITH
    REM SPECIAL CASE
142: LET Multi=0
144: LET T=0.52
146: LET UNIT=0
148: REM CULD MAKE MORE SPIN IN
    REM 1000000000
150: REM SETUP, N
152: REM SETUP, M
154: REM SETUP, U
156: REM CALL LAB, T
158: REM FAST
160: REM MAKE LEDOUT
162: GOTO
164: RETURN
    REM. END

```

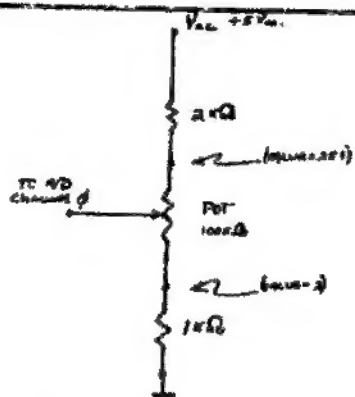
Fig. 3. Δ and Δ (continued)

```

1  REM      (M12 IS A MACRO ASSEMBLER)
2
3  FAST
4  REM  BASIL LEADOUT
5  11 LET M=0
6  12 LET P=0
7  13 LET V=0
8  14 LET OUTOUT=10000
9  15 IF (M+P+V)>13
10  20 DIM V(4)
11
12 REM MAIN
13 100 SLW
14 110 PRINT "INPUT DECIMAL VALUE"
15 120 INPUT D
16 130 PRINT D
17 140 LET M=64-INT (D/100)
18 150 LET P=32-INT (D-(M+64)*100)
19 160
20 170 LET V(0)=INT (D-(M+64)*100-
21 180 -P)/100
22 190 LET V(1)=M
23 200 LET V(2)=P
24 210 LET V(3)=0
25 220 LET V(4)=0
26 230 FOR J=1 TO 50
27 240 FOR I=1 TO 3
28 250 GOSUB SUB(V(I))
29 260 FAST
30 270 LET A=0.5 OUTOUT
31 280 NEXT J
32 290 NEXT I
33 300 NEXT J
34 310 FOR Q=V(4)
35 320 LET A=V(4)-THEN GOTO 100
36 330 STOP
37
38 REM SUBROUT
39 400 FOR K=10511,0
40 410 FAST
41 420 LET L=INT 10000
42 430 LET M=INT 10511
43 440 SLW
44 450 GOTO 115
45 460 STOP

```

Fig. 4
Pure BASIC LEADERS
(w/ Auto-B)



```

3099 REM #D7E7
3100 FAST
3110 REM T=1 TO 10
3120 FOR K=1 TO T
3130 LET L=USR10000
3140 PRINT "VALUE=";P=LN 10/5
3150 NEXT K
3160 PC=LN 10/5

```

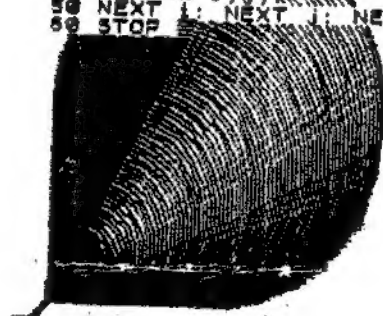
7-10-68, 17000, 18000, 19000, 20000

1942. 12. 25
 1943. 1. 1
 1943. 1. 2
 1943. 1. 3
 1943. 1. 4
 1943. 1. 5
 1943. 1. 6
 1943. 1. 7
 1943. 1. 8
 1943. 1. 9
 1943. 1. 10

```

10 FOR F=1 TO 10
20 FOR J=1 TO 10
30 FOR K=1 TO 100
40 CIRCLE I,J,K
50 NEXT K
60 NEXT J
70 NEXT F

```



That's wrong with this program

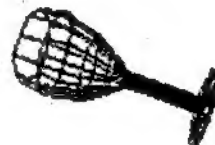
SALVENDY/GUP-TSIT

It was 12/78 keyboard started acting up. It was the because of your working between the keyboard's flat ribbon connector and the keyboard. I had another 3 days on the claim then the assembly was no longer possible. I tried look at the 12/80 schematics show these keys (B, I, M, ...) to be associated with the 20-015 line on the 12 connector block. Apparently, this may also work this on the plastic carrier can be used as off during assembly.

Roll the picture out of their jaws and hold the ends up to a light. If the
entire film was stretched off, you'll see daylight through the screen. Repair
is simple, just take a sharp ^{sharp} pointed scissors and trim closely 1/8" from the
bottom bottom edge of the film until you've tapered the mouth.

Improving the problem is just as easy - If you leave volume (e.g., V, S, P, S) and δ , γ , β , you've lost contact with one of the CBC lines in this case (2%); in the 9 pct remaining, it seems now best for us to cancel; you'll know 9 days, so suspect the right old time.

Sample number 1705 W-10



WHAT A CHARACTER

The Game Changer Interface (reviewed elsewhere) enables you to upload AXARI VCS (2600) cartridge contents into your ZX/TS, SAVE them on tape and play 'em, using the ZX/TS RAM. Since the copy of the game is in RAM, you can also change any part you wish, to suit your own personal style.

The AXARI uses a 6302 (processor)/6301 (significant changes require programming in 6302 machine code).

However, an understanding of 6302 machine code (MC) is not really necessary for you to change some aspects of your recorded AXARI game. Specifically, any character or icon (a symbol representing you, an object, enemy etc.) can be changed to another shape, just by FORKING new values into its data array.

In order to be represented on the screen, an object must have "bits" stored somewhere in memory. The game software reads these bits to the hardware which in turn produces the picture you see on the screen. Here's a simple example of a face using an 8 X 8 bit image (that is, 8 bits wide by 8 bits high):

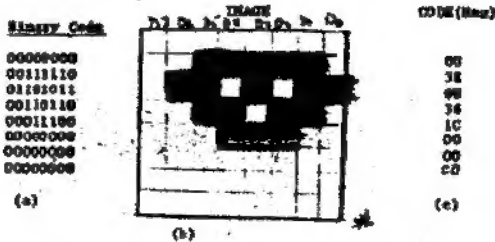


Fig. 1

Figure 1(a) gives the binary representation of our "face". Here ones (1's) will represent "turned on" bits and will be dark, while 0's are turned off or blank (the background color will show through). In Fig. 1(c), since this image is exactly 8 bits wide, we've shown the hexadecimal numbers representing one bit byte. This face can be stored in 8 consecutive memory locations.

If we were to pull up the AXARI game code on the screen a byte at a time and represent it as a series of black and white squares we could "see" the actual character images used in the game. Fig. 2 gives a partial program listing that will do just that.

Before you GOTO 7000 however, you may have to do a little detective work. A typical AXARI cartridge contains a 4K ROM which you will already have downloaded to the 4K of RAM starting at 18314. The 6302 interrupts (where the program restarts after something important or bad happens) point to the very high memory locations and addresses pointing to the actual program code will be stored there. You can use the disassembler or monitor to check the top sections of RAM (7774 through 7777) for these pointers. You can be reasonably sure the data tables for the characters are not in the immediate areas pointed to by these three sets of 3 bytes and should look elsewhere. Places to look are: 1) boundaries between pages (7000, 7200 etc. - a "page" is 256 bytes) 2) the beginning of the program 3) addresses which occur frequently.

For AXARI BASIC we used the disassembler to spot check some of these areas and found absolute garbage at the beginning of the program. A good hard look at the 1 K (see Fig 3) also shows the experienced programmer something which just about had to be someone. The recurrence of the same code number several times in succession is very unlikely in an MC program. These clues led us to choose 18314 as a starting address. After offset, this address corresponds to 7000(10) in the AXARI.

After going to 7000 we obtained the listing in Fig. 4. A pattern of dots is clearly emerging, but hard to identify as letters or figures. Connecting the dots (or using the inverse space instead of an asterisk) can help make the patterns more intelligible. We've done that for the "GOTO" command of AXARI BASIC. You should be able to pick out some other upside-down commands like PRINT, and ELSE. The game software and hardware automatically turn these right side up.

Now does the routine work? First we disassemble some new variables and pick a starting address. These are:

- S = Starting address (decimal only)
- D(1) = The binary 1 th digit of a byte
- A\$ = A string which will print either filled-in or MC squares
- F = The decimal value of the byte at location S
- N = A temporary "part" of the F variable as it is reduced to its 8 or binary digits.

Next we PRINT the data stored in the currently chosen memory location. The subroutine at 8090 repeatedly divides the decimal number by 2, saving the remainder as D(J) to convert the decimal number to its binary (actually binary will stored as decimal) bit equivalents.

Returning to 70125 from the subroutine we convert each "character" in the string A\$ to its bit pattern; it remains a blank if D(J)=1 (that is, if its 4) and becomes an asterisk if D(J)=1.

Finally we print the memory location, decimal value and our new blocked out string of "bits". Don't forget that the actual addresses where this data table resides are some 12000 (decimal) memory locations higher in the AXARI. Also, the conventions used by this program may not be used in other games. In AXARI BASIC, in fact, two more, different, techniques are used for addressing the numbers (for NAME) and the display area (time).

You should be able to add shape lines to the "change" program with no problems on 4K or 16K games. You could even just pick an address at random and observe the screen for intelligible patterns. For MC games create a from standing

version of the program addition and try to FIND the (with the actual cartridge installed) memory locations above 12000. Remember, the bank switcher may get in the way.

Some hints on what to change in AXARI BASIC:

- 1) The symbol table for the display area (START, PROGRAM etc.) Abbreviating them would free up RAM for your own 6302 MC routines.
- 2) Change the "Block" graphics character to more interesting pictures (e.g., a smiley face).
- 3) Change the alpha or numeric characters to provide a more interesting display. The letters, for example, can be converted to those of another language.
- 4) Your own ideas.

```

7000:GOTO 8100
7002:GOTO 8400
7004:PRINT " INPUT STARTING ADDR"
7006: "
7008:INPUT S
7010:FOR I=0 TO 7:GOTO 7012
7012:LET F=PEEK S
7014:GOTO 7016
7016:FOR J=0 TO 7
7018:LET D(J)=F/2
7020:LET F=F/2
7022:IF D(J)=1 THEN LET A$=A$+"*"
7024:IF D(J)=0 THEN LET A$=A$+" "
7026:NEXT J
7028:PRINT S: " "
7030:GOTO 7008
7032: "
7034:PRINT "
7036:GOTO 7000
7038: "
7040: "
7042: "
7044: "
7046: "
7048: "
7050: "
7052: "
7054: "
7056: "
7058: "
7060: "
7062: "
7064: "
7066: "
7068: "
7070: "
7072: "
7074: "
7076: "
7078: "
7080: "
7082: "
7084: "
7086: "
7088: "
7090: "
7092: "
7094: "
7096: "
7098: "
7100: "
7102: "
7104: "
7106: "
7108: "
7110: "
7112: "
7114: "
7116: "
7118: "
7120: "
7122: "
7124: "
7126: "
7128: "
7130: "
7132: "
7134: "
7136: "
7138: "
7140: "
7142: "
7144: "
7146: "
7148: "
7150: "
7152: "
7154: "
7156: "
7158: "
7160: "
7162: "
7164: "
7166: "
7168: "
7170: "
7172: "
7174: "
7176: "
7178: "
7180: "
7182: "
7184: "
7186: "
7188: "
7190: "
7192: "
7194: "
7196: "
7198: "
7200: "
7202: "
7204: "
7206: "
7208: "
7210: "
7212: "
7214: "
7216: "
7218: "
7220: "
7222: "
7224: "
7226: "
7228: "
7230: "
7232: "
7234: "
7236: "
7238: "
7240: "
7242: "
7244: "
7246: "
7248: "
7250: "
7252: "
7254: "
7256: "
7258: "
7260: "
7262: "
7264: "
7266: "
7268: "
7270: "
7272: "
7274: "
7276: "
7278: "
7280: "
7282: "
7284: "
7286: "
7288: "
7290: "
7292: "
7294: "
7296: "
7298: "
7300: "
7302: "
7304: "
7306: "
7308: "
7310: "
7312: "
7314: "
7316: "
7318: "
7320: "
7322: "
7324: "
7326: "
7328: "
7330: "
7332: "
7334: "
7336: "
7338: "
7340: "
7342: "
7344: "
7346: "
7348: "
7350: "
7352: "
7354: "
7356: "
7358: "
7360: "
7362: "
7364: "
7366: "
7368: "
7370: "
7372: "
7374: "
7376: "
7378: "
7380: "
7382: "
7384: "
7386: "
7388: "
7390: "
7392: "
7394: "
7396: "
7398: "
7400: "
7402: "
7404: "
7406: "
7408: "
7410: "
7412: "
7414: "
7416: "
7418: "
7420: "
7422: "
7424: "
7426: "
7428: "
7430: "
7432: "
7434: "
7436: "
7438: "
7440: "
7442: "
7444: "
7446: "
7448: "
7450: "
7452: "
7454: "
7456: "
7458: "
7460: "
7462: "
7464: "
7466: "
7468: "
7470: "
7472: "
7474: "
7476: "
7478: "
7480: "
7482: "
7484: "
7486: "
7488: "
7490: "
7492: "
7494: "
7496: "
7498: "
7500: "
7502: "
7504: "
7506: "
7508: "
7510: "
7512: "
7514: "
7516: "
7518: "
7520: "
7522: "
7524: "
7526: "
7528: "
7530: "
7532: "
7534: "
7536: "
7538: "
7540: "
7542: "
7544: "
7546: "
7548: "
7550: "
7552: "
7554: "
7556: "
7558: "
7560: "
7562: "
7564: "
7566: "
7568: "
7570: "
7572: "
7574: "
7576: "
7578: "
7580: "
7582: "
7584: "
7586: "
7588: "
7590: "
7592: "
7594: "
7596: "
7598: "
7600: "
7602: "
7604: "
7606: "
7608: "
7610: "
7612: "
7614: "
7616: "
7618: "
7620: "
7622: "
7624: "
7626: "
7628: "
7630: "
7632: "
7634: "
7636: "
7638: "
7640: "
7642: "
7644: "
7646: "
7648: "
7650: "
7652: "
7654: "
7656: "
7658: "
7660: "
7662: "
7664: "
7666: "
7668: "
7670: "
7672: "
7674: "
7676: "
7678: "
7680: "
7682: "
7684: "
7686: "
7688: "
7690: "
7692: "
7694: "
7696: "
7698: "
7700: "
7702: "
7704: "
7706: "
7708: "
7710: "
7712: "
7714: "
7716: "
7718: "
7720: "
7722: "
7724: "
7726: "
7728: "
7730: "
7732: "
7734: "
7736: "
7738: "
7740: "
7742: "
7744: "
7746: "
7748: "
7750: "
7752: "
7754: "
7756: "
7758: "
7760: "
7762: "
7764: "
7766: "
7768: "
7770: "
7772: "
7774: "
7776: "
7778: "
7780: "
7782: "
7784: "
7786: "
7788: "
7790: "
7792: "
7794: "
7796: "
7798: "
7800: "
7802: "
7804: "
7806: "
7808: "
7810: "
7812: "
7814: "
7816: "
7818: "
7820: "
7822: "
7824: "
7826: "
7828: "
7830: "
7832: "
7834: "
7836: "
7838: "
7840: "
7842: "
7844: "
7846: "
7848: "
7850: "
7852: "
7854: "
7856: "
7858: "
7860: "
7862: "
7864: "
7866: "
7868: "
7870: "
7872: "
7874: "
7876: "
7878: "
7880: "
7882: "
7884: "
7886: "
7888: "
7890: "
7892: "
7894: "
7896: "
7898: "
7900: "
7902: "
7904: "
7906: "
7908: "
7910: "
7912: "
7914: "
7916: "
7918: "
7920: "
7922: "
7924: "
7926: "
7928: "
7930: "
7932: "
7934: "
7936: "
7938: "
7940: "
7942: "
7944: "
7946: "
7948: "
7950: "
7952: "
7954: "
7956: "
7958: "
7960: "
7962: "
7964: "
7966: "
7968: "
7970: "
7972: "
7974: "
7976: "
7978: "
7980: "
7982: "
7984: "
7986: "
7988: "
7990: "
7992: "
7994: "
7996: "
7998: "
8000: "
8002: "
8004: "
8006: "
8008: "
8010: "
8012: "
8014: "
8016: "
8018: "
8020: "
8022: "
8024: "
8026: "
8028: "
8030: "
8032: "
8034: "
8036: "
8038: "
8040: "
8042: "
8044: "
8046: "
8048: "
8050: "
8052: "
8054: "
8056: "
8058: "
8060: "
8062: "
8064: "
8066: "
8068: "
8070: "
8072: "
8074: "
8076: "
8078: "
8080: "
8082: "
8084: "
8086: "
8088: "
8090: "
8092: "
8094: "
8096: "
8098: "
8100: "
8102: "
8104: "
8106: "
8108: "
8110: "
8112: "
8114: "
8116: "
8118: "
8120: "
8122: "
8124: "
8126: "
8128: "
8130: "
8132: "
8134: "
8136: "
8138: "
8140: "
8142: "
8144: "
8146: "
8148: "
8150: "
8152: "
8154: "
8156: "
8158: "
8160: "
8162: "
8164: "
8166: "
8168: "
8170: "
8172: "
8174: "
8176: "
8178: "
8180: "
8182: "
8184: "
8186: "
8188: "
8190: "
8192: "
8194: "
8196: "
8198: "
8200: "
8202: "
8204: "
8206: "
8208: "
8210: "
8212: "
8214: "
8216: "
8218: "
8220: "
8222: "
8224: "
8226: "
8228: "
8230: "
8232: "
8234: "
8236: "
8238: "
8240: "
8242: "
8244: "
8246: "
8248: "
8250: "
8252: "
8254: "
8256: "
8258: "
8260: "
8262: "
8264: "
8266: "
8268: "
8270: "
8272: "
8274: "
8276: "
8278: "
8280: "
8282: "
8284: "
8286: "
8288: "
8290: "
8292: "
8294: "
8296: "
8298: "
8300: "
8302: "
8304: "
8306: "
8308: "
8310: "
8312: "
8314: "
8316: "
8318: "
8320: "
8322: "
8324: "
8326: "
8328: "
8330: "
8332: "
8334: "
8336: "
8338: "
8340: "
8342: "
8344: "
8346: "
8348: "
8350: "
8352: "
8354: "
8356: "
8358: "
8360: "
8362: "
8364: "
8366: "
8368: "
8370: "
8372: "
8374: "
8376: "
8378: "
8380: "
8382: "
8384: "
8386: "
8388: "
8390: "
8392: "
8394: "
8396: "
8398: "
8400: "
8402: "
8404: "
8406: "
8408: "
8410: "
8412: "
8414: "
8416: "
8418: "
8420: "
8422: "
8424: "
8426: "
8428: "
8430: "
8432: "
8434: "
8436: "
8438: "
8440: "
8442: "
8444: "
8446: "
8448: "
8450: "
8452: "
8454: "
8456: "
8458: "
8460: "
8462: "
8464: "
8466: "
8468: "
8470: "
8472: "
8474: "
8476: "
8478: "
8480: "
8482: "
8484: "
8486: "
8488: "
8490: "
8492: "
8494: "
8496: "
8498: "
8500: "
8502: "
8504: "
8506: "
8508: "
8510: "
8512: "
8514: "
8516: "
8518: "
8520: "
8522: "
8524: "
8526: "
8528: "
8530: "
8532: "
8534: "
8536: "
8538: "
8540: "
8542: "
8544: "
8546: "
8548: "
8550: "
8552: "
8554: "
8556: "
8558: "
8560: "
8562: "
8564: "
8566: "
8568: "
8570: "
8572: "
8574: "
8576: "
8578: "
8580: "
8582: "
8584: "
8586: "
8588: "
8590: "
8592: "
8594: "
8596: "
8598: "
8600: "
8602: "
8604: "
8606: "
8608: "
8610: "
8612: "
8614: "
8616: "
8618: "
8620: "
8622: "
8624: "
8626: "
8628: "
8630: "
8632: "
8634: "
8636: "
8638: "
8640: "
8642: "
8644: "
8646: "
8648: "
8650: "
8652: "
8654: "
8656: "
8658: "
8660: "
8662: "
8664: "
8666: "
8668: "
8670: "
8672: "
8674: "
8676: "
8678: "
8680: "
8682: "
8684: "
8686: "
8688: "
8690: "
8692: "
8694: "
8696: "
8698: "
8700: "
8702: "
8704: "
8706: "
8708: "
8710: "
8712: "
8714: "
8716: "
8718: "
8720: "
8722: "
8724: "
8726: "
8728: "
8730: "
8732: "
8734: "
8736: "
8738: "
8740: "
8742: "
8744: "
8746: "
8748: "
8750: "
8752: "
8754: "
8756: "
8758: "
8760: "
8762: "
8764: "
8766: "
8768: "
8770: "
8772: "
8774: "
8776: "
8778: "
8780: "
8782: "
8784: "
8786: "
8788: "
8790: "
8792: "
8794: "
8796: "
8798: "
8800: "
8802: "
8804: "
8806: "
8808: "
8810: "
8812: "
8814: "
8816: "
8818: "
8820: "
8822: "
8824: "
8826: "
8828: "
8830: "
8832: "
8834: "
8836: "
8838: "
8840: "
8842: "
8844: "
8846: "
8848: "
8850: "
8852: "
8854: "
8856: "
8858: "
8860: "
8862: "
8864: "
8866: "
8868: "
8870: "
8872: "
8874: "
8876: "
8878: "
8880: "
8882: "
8884: "
8886: "
8888: "
8890: "
8892: "
8894: "
8896: "
8898: "
8900: "
8902: "
8904: "
8906: "
8908: "
8910: "
8912: "
8914: "
8916: "
8918: "
8920: "
8922: "
8924: "
8926: "
8928: "
8930: "
8932: "
8934: "
8936: "
8938: "
8940: "
8942: "
8944: "
8946: "
8948: "
8950: "
8952: "
8954: "
8956: "
8958: "
8960: "
8962: "
8964: "
8966: "
8968: "
8970: "
8972: "
8974: "
8976: "
8978: "
8980: "
8982: "
8984: "
8986: "
8988: "
8990: "
8992: "
8994: "
8996: "
8998: "
9000: "
9002: "
9004: "
9006: "
9008: "
9010: "
9012: "
9014: "
9016: "
9018: "
9020: "
9022: "
9024: "
9026: "
9028: "
9030: "
9032: "
9034: "
9036: "
9038: "
9040: "
9042: "
9044: "
9046: "
9048: "
9050: "
9052: "
9054: "
9056: "
9058: "
9060: "
9062: "
9064: "
9066: "
9068: "
9070: "
9072: "
9074: "
9076: "
9078: "
9080: "
9082: "
9084: "
9086: "
9088: "
9090: "
9092: "
9094: "
9096: "
9098: "
9100: "
9102: "
9104: "
9106: "
9108: "
9110: "
9112: "
9114: "
9116: "
9118: "
9120: "
9122: "
9124: "
9126: "
9128: "
9130: "
9132: "
9134: "
9136: "
9138: "
9140: "
9142: "
9144: "
9146: "
9148: "
9150: "
9152: "
9154: "
9156: "
9158: "
9160: "
9162: "
9164: "
9166: "
9168: "
9170: "
9172: "
9174: "
9176: "
9178: "
9180: "
9182: "
9184: "
9186: "
9188: "
9190: "
9192: "
9194: "
9196: "
9198: "
9200: "
9202: "
9204: "
9206: "
9208: "
9210: "
9212: "
9214: "
9216: "
9218: "
9220: "
9222: "
9224: "
9226: "
9228: "
9230: "
9232: "
9234: "
9236: "
9238: "
9240: "
9242: "
9244: "
9246: "
9248: "
9250: "
9252: "
9254: "
9256: "
9258: "
9260: "
9262: "
9264: "
9266: "
9268: "
9270: "
9272: "
9274: "
9276: "
9278: "
9280: "
9282: "
9284: "
9286: "
9288: "
9290: "
9292: "
9294: "
9296: "
9298: "
9300: "
9302: "
9304: "
9306: "
9308: "
9310: "
9312: "
9314: "
9316: "
9318: "
9320: "
9322: "
9324: "
9326: "
9328: "
9330: "
9332: "
9334: "
9336: "
9338: "
9340: "
9342: "
9344: "
9346: "
9348: "
9350: "
9352: "
9354: "
9356: "
9358: "
9360: "
9362: "
9364: "
9366: "
9368: "
9370: "
9372: "
9374: "
9376: "
9378: "
9380: "
9382: "
9384: "
9386: "
9388: "
9390: "
9392: "
9394: "
9396: "
9398: "
9400: "
9402: "
9404: "
9406: "
9408: "
9410: "
9412: "
9414: "
9416: "
9418: "
9420: "
9422: "
9424: "
9426: "
9428: "
9430: "
9432: "
9434: "
9436: "
9438: "
9440: "
9442: "
9444: "
9446: "
9448: "
9450: "
9452: "
9454: "
9456: "
9458: "
9460: "
9462: "
9464: "
9466: "
9468: "
9470: "
9472: "
9474: "
9476: "
9478: "
9480: "
9482: "
9484: "
9486: "
9488: "
9490: "
9492: "
9494: "
9496: "
9498: "
9500: "
9502: "
9504: "
9506: "
9508: "
9510: "
9512: "
9514: "
9516: "
9518: "
9520: "
9522: "
9524: "
9526: "
9528: "
9530: "
9532: "
9534: "
9536: "
9538: "
9540: "
9542: "
9544: "
9546: "
9548: "
9550: "
9552: "
9554: "
9556: "
9558: "
9560: "
9562: "
9564: "
9566: "
9568: "
9570: "
9572: "
9574: "
9576: "
9578: "
9580: "
9582: "
9584: "
9586: "
9588: "
9590: "
9592: "
9594: "
9596: "
9598: "
9600: "
9602: "
9604: "
9606: "
9608: "
9610: "
9612: "
9614: "
9616: "
9618: "
9620: "
9622: "
9624: "
9626: "
9628: "
9630: "
9632: "
9634: "
9636: "
9638: "
9640: "
9642: "
9644: "
9646: "
9648: "
9650: "
9652: "
9654: "
9656: "
9658: "
9660: "
9662: "
9664: "
9666: "
9668: "
9670: "
9672: "
9674: "
9676: "
9678: "
9680: "
9682: "
9684: "
9686: "
9688: "
9690: "
9692: "
9694: "
9696: "
9698: "
9700: "
9702: "
9704: "
9706: "
9708: "
9710: "
9712: "
9714: "
9716: "
9718: "
9720: "
9722: "
9724: "
9726: "
9728: "
9730: "
9732: "
9734: "
9736: "
9738: "
9740: "
9742: "
9744: "
9746: "
9748: "
9750: "
9752: "
9754: "
9756: "
9758: "
9760: "
9762: "
9764: "
9766: "
9768: "
9770: "
9772: "
9774: "
9776: "
9778: "
9780: "
9782: "
9784: "
9786: "
9788: "
9790: "
9792: "
9794: "
9796: "
9798: "
9800: "
9802: "
9804: "
9806: "
9808: "
9810: "
9812: "
9814: "
9816: "
9818: "
9820: "
9822: "
9824: "
9826: "
9828: "
9830: "
9832: "
9834: "
9836: "
9838: "
9840: "
9842: "
9844: "
9846: "
9848: "
9850: "
9852: "
9
```

DRIVING THE LED'S

In any number of monitoring and/or control uses for your EX/TS, you may find that using a TV set to check on your system is inconvenient or impractical. This would be so, for example, in outdoor or dirty environments or when you simply wish to check the status of one or two system parameters without the bother of hooking up a TV or monitor. One fairly easy and inexpensive way to do this is to use seven-segment LED (light emitting diode) displays on your system computer. Your EX/TS bus doesn't have enough power to directly drive these displays, but a few simple IC's and an I/O board can give you this capability.

You must have seen one of 8 bit output ports. I used Star-2's Report Generator board, but JE Amis's I/O and even Ryan Buck's RS-1 module (without the relays) should be suitable as a buffered output from your computer. The only other hardware you'll need, aside from miscellaneous wire and some resistors, is a 7416, bus inverting buffers, a 7411 BCD (Binary Coded Decimal)-to-7 segment decoder/driver, and a 3 character, 7 segment LED display (MAG). A schematic of this simple circuit is given in Fig 1.

This circuit represents a "middle ground" approach to producing a 3 digit output display. More hardware (latches etc.) could have been used, resulting in permanently lit displays, or we could have used software to create a controlling outer loop to keep the monitoring function as the main operating system. The combination of hardware and software I chose represents a system which uses multiplexing to provide the appearance of a full three digit display.

As you can see from Figure one, we are using the EX's data lines D₀ through D₇ to send out the binary coded decimal value of one of three digits (units, tens or hundreds). The current digit is determined by the data lines D₀ through D₆ which complete the path to ground for the appropriate display. Line D₇ enables and latches the last value from lines D₀ to D₆ into the 7411 which in turn, drives the appropriate elements of the display. Multiplexing is provided by the M software driver routine shown in Fig 2.

After data is POKE'd into the three buffer locations (fig 2-a) the M program outputs the specially constructed character code for each digit in sequence. This operation occurs so rapidly that the eye is fooled into thinking each digit is lit all the time. If you'd like to see what's really happening, you can either increase the dwell time between digits or use the BASIC listing in fig 3. The BASIC listing requires no machine code but is, of course, slow.

The software is applicable to the Star-2 Report Generator board as it stands. However, you can change the address of the output 16551(c) to the address of your own OUTPUT subroutine. The Report Generator uses a PEO and is I/O capped. For memory capped systems, your OUTPUT subroutine would probably consist of simply putting the value in the accumulator, loading the designated memory location with the value, and returning to the driver.

I built this circuit on a small ACE (all circuit evaluator) board and obtained its power from the +5 volt regulator on the R.O. board. The values for the current limiting resistors are not critical, but should be as large as possible to conserve power, while still giving adequate brightness. One's overlook the 10K pullup resistors for the 7416, or forget to tie the unused inputs to ground.

Fig 3b gives a BASIC driver which I use to obtain a temperature reading from the A/D converter on the R.O. board and output that value to the LED display. The code shown would actually be a subroutine called from a main menu. The display is normally blank. By touching, say, "Y" on my keyboard, I would be selecting a menu item which calls the Temperature output routine. All this can be done without the need of a TV screen, if I know in advance (and I do) which key to press. The system could be expanded, with more chips, lots of software and perhaps even LCD displays to actually let you program your computer without a TV. A modification here would be to use hex buffer drivers instead of BCD.

Note too, that I used "junk box" parts from around my shop to build this particular display. A proper design would use perhaps a 74123 as the latch driver (pinout is the same as the 7411). Also remember that the output codes which will vary on all three digits at one time (e.g., code 01111111, produces all 8's), on current drive will be very high.

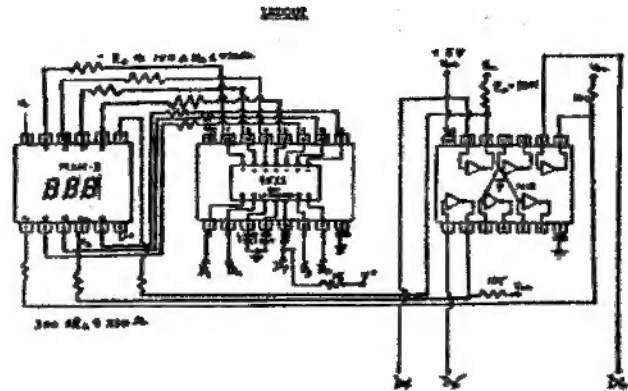


FIG 1 (a)

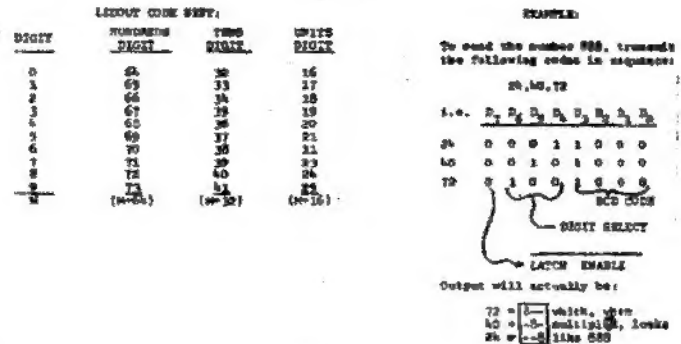


Fig 1 (b)

NAME	ADDRESS	DATA	FUNCTION	COMMENT	SIZE
LEADOUT	16540	byte	LEAD, 00FF	Send 0 of codes	00000
LEAD	16541	byte	LEAD, 00FF	Send 0 of codes	00000
LEAD	16542	byte	LEAD, 00FF	Send 0 of codes	00000
LEAD	16543	byte	LEAD, 00FF	Send 0 of codes	00000
LEAD	16544	byte	LEAD, 00FF	Send 0 of codes	00000
LEAD	16545	byte	LEAD, 00FF	Send 0 of codes	00000
LEAD	16546	byte	LEAD, 00FF	Send 0 of codes	00000
LEAD	16547	byte	LEAD, 00FF	Send 0 of codes	00000
LEAD	16548	byte	LEAD, 00FF	Send 0 of codes	00000
LEAD	16549	byte	LEAD, 00FF	Send 0 of codes	00000
LEAD	16550	byte	LEAD, 00FF	Send 0 of codes	00000
LEAD	16551	byte	LEAD, 00FF	Send 0 of codes	00000
LEAD	16552	byte	LEAD, 00FF	Send 0 of codes	00000
LEAD	16553	byte	LEAD, 00FF	Send 0 of codes	00000
LEAD	16554	byte	LEAD, 00FF	Send 0 of codes	00000
LEAD	16555	byte	LEAD, 00FF	Send 0 of codes	00000
LEAD	16556	byte	LEAD, 00FF	Send 0 of codes	00000
LEAD	16557	byte	LEAD, 00FF	Send 0 of codes	00000
LEAD	16558	byte	LEAD, 00FF	Send 0 of codes	00000
LEAD	16559	byte	LEAD, 00FF	Send 0 of codes	00000
LEAD	16560	byte	LEAD, 00FF	Send 0 of codes	00000
LEAD	16561	byte	LEAD, 00FF	Send 0 of codes	00000
LEAD	16562	byte	LEAD, 00FF	Send 0 of codes	00000
LEAD	16563	byte	LEAD, 00FF	Send 0 of codes	00000
LEAD	16564	byte	LEAD, 00FF	Send 0 of codes	00000
LEAD	16565	byte	LEAD, 00FF	Send 0 of codes	00000
LEAD	16566	byte	LEAD, 00FF	Send 0 of codes	00000
LEAD	16567	byte	LEAD, 00FF	Send 0 of codes	00000
LEAD	16568	byte	LEAD, 00FF	Send 0 of codes	00000
LEAD	16569	byte	LEAD, 00FF	Send 0 of codes	00000
LEAD	16570	byte	LEAD, 00FF	Send 0 of codes	00000
LEAD	16571	byte	LEAD, 00FF	Send 0 of codes	00000
LEAD	16572	byte	LEAD, 00FF	Send 0 of codes	00000
LEAD	16573	byte	LEAD, 00FF	Send 0 of codes	00000
LEAD	16574	byte	LEAD, 00FF	Send 0 of codes	00000
LEAD	16575	byte	LEAD, 00FF	Send 0 of codes	00000
LEAD	16576	byte	LEAD, 00FF	Send 0 of codes	00000
LEAD	16577	byte	LEAD, 00FF	Send 0 of codes	00000
LEAD	16578	byte	LEAD, 00FF	Send 0 of codes	00000
LEAD	16579	byte	LEAD, 00FF	Send 0 of codes	00000
LEAD	16580	byte	LEAD, 00FF	Send 0 of codes	00000
LEAD	16581	byte	LEAD, 00FF	Send 0 of codes	00000
LEAD	16582	byte	LEAD, 00FF	Send 0 of codes	00000
LEAD	16583	byte	LEAD, 00FF	Send 0 of codes	00000
LEAD	16584	byte	LEAD, 00FF	Send 0 of codes	00000
LEAD	16585	byte	LEAD, 00FF	Send 0 of codes	00000
LEAD	16586	byte	LEAD, 00FF	Send 0 of codes	00000
LEAD	16587	byte	LEAD, 00FF	Send 0 of codes	00000
LEAD	16588	byte	LEAD, 00FF	Send 0 of codes	00000
LEAD	16589	byte	LEAD, 00FF	Send 0 of codes	00000
LEAD	16590	byte	LEAD, 00FF	Send 0 of codes	00000
LEAD	16591	byte	LEAD, 00FF	Send 0 of codes	00000
LEAD	16592	byte	LEAD, 00FF	Send 0 of codes	00000
LEAD	16593	byte	LEAD, 00FF	Send 0 of codes	00000
LEAD	16594	byte	LEAD, 00FF	Send 0 of codes	00000
LEAD	16595	byte	LEAD, 00FF	Send 0 of codes	00000
LEAD	16596	byte	LEAD, 00FF	Send 0 of codes	00000
LEAD	16597	byte	LEAD, 00FF	Send 0 of codes	00000
LEAD	16598	byte	LEAD, 00FF	Send 0 of codes	00000
LEAD	16599	byte	LEAD, 00FF	Send 0 of codes	00000
LEAD	16600	byte	LEAD, 00FF	Send 0 of codes	00000
LEAD	16601	byte	LEAD, 00FF	Send 0 of codes	00000
LEAD	16602	byte	LEAD, 00FF	Send 0 of codes	00000
LEAD	16603	byte	LEAD, 00FF	Send 0 of codes	00000
LEAD	16604	byte	LEAD, 00FF	Send 0 of codes	00000
LEAD	16605	byte	LEAD, 00FF	Send 0 of codes	00000
LEAD	16606	byte	LEAD, 00FF	Send 0 of codes	00000
LEAD	16607	byte	LEAD, 00FF	Send 0 of codes	00000
LEAD	16608	byte	LEAD, 00FF	Send 0 of codes	00000
LEAD	16609	byte	LEAD, 00FF	Send 0 of codes	00000
LEAD	16610	byte	LEAD, 00FF	Send 0 of codes	00000
LEAD	16611	byte	LEAD, 00FF	Send 0 of codes	00000
LEAD	16612	byte	LEAD, 00FF	Send 0 of codes	00000
LEAD	16613	byte	LEAD, 00FF	Send 0 of codes	00000
LEAD	16614	byte	LEAD, 00FF	Send 0 of codes	00000
LEAD	16615	byte	LEAD, 00FF	Send 0 of codes	00000
LEAD	16616	byte	LEAD, 00FF	Send 0 of codes	00000
LEAD	16617	byte	LEAD, 00FF	Send 0 of codes	00000
LEAD	16618	byte	LEAD, 00FF	Send 0 of codes	00000
LEAD	16619	byte	LEAD, 00FF	Send 0 of codes	00000
LEAD	16620	byte	LEAD, 00FF	Send 0 of codes	00000
LEAD	16621	byte	LEAD, 00FF	Send 0 of codes	00000
LEAD	16622	byte	LEAD, 00FF	Send 0 of codes	00000
LEAD	16623	byte	LEAD, 00FF	Send 0 of codes	00000
LEAD	16624	byte	LEAD, 00FF	Send 0 of codes	00000
LEAD	16625	byte	LEAD, 00FF	Send 0 of codes	00000
LEAD	16626	byte	LEAD, 00FF	Send 0 of codes	00000
LEAD	16627	byte	LEAD, 00FF	Send 0 of codes	00000
LEAD	16628	byte	LEAD, 00FF	Send 0 of codes	00000
LEAD	16629	byte	LEAD, 00FF	Send 0 of codes	00000
LEAD	16630	byte	LEAD, 00FF	Send 0 of codes	00000
LEAD	16631	byte	LEAD, 00FF	Send 0 of codes	00000
LEAD	16632	byte	LEAD, 00FF	Send 0 of codes	00000
LEAD	16633	byte	LEAD, 00FF	Send 0 of codes	00000
LEAD	16634	byte	LEAD, 00FF	Send 0 of codes	00000
LEAD	16635	byte	LEAD, 00FF	Send 0 of codes	00000
LEAD	16636	byte	LEAD, 00FF	Send 0 of codes	00000
LEAD	16637	byte	LEAD, 00FF	Send 0 of codes	00000
LEAD	16638	byte	LEAD, 00FF	Send 0 of codes	00000
LEAD	16639	byte	LEAD, 00FF	Send 0 of codes	00000
LEAD	16640	byte	LEAD, 00FF	Send 0 of codes	00000
LEAD	16641	byte	LEAD, 00FF	Send 0 of codes	00000
LEAD	16642	byte	LEAD, 00FF	Send 0 of codes	00000
LEAD	16643	byte	LEAD, 00FF	Send 0 of codes	00000
LEAD	16644	byte	LEAD, 00FF	Send 0 of codes	00000
LEAD	16645	byte	LEAD, 00FF	Send 0 of codes	00000
LEAD	16646	byte	LEAD, 00FF	Send 0 of codes	00000
LEAD	16647	byte	LEAD, 00FF	Send 0 of codes	00000
LEAD	16648	byte	LEAD, 00FF	Send 0 of codes	00000
LEAD	16649	byte	LEAD, 00FF	Send 0 of codes	00000
LEAD	16650	byte	LEAD, 00FF	Send 0 of codes	00000
LEAD	16651	byte	LEAD, 00FF	Send 0 of codes	00000
LEAD	16652	byte	LEAD, 00FF	Send 0 of codes	00000
LEAD	16653	byte	LEAD, 00FF	Send 0 of codes	00000
LEAD	16654	byte	LEAD, 00FF	Send 0 of codes	00000
LEAD	16655	byte	LEAD, 00FF	Send 0 of codes	00000
LEAD	16656	byte	LEAD, 00FF	Send 0 of codes	00000
LEAD	16657	byte	LEAD, 00FF	Send 0 of codes	00000
LEAD	16658	byte	LEAD, 00FF	Send 0 of codes	00000
LEAD	16659	byte	LEAD, 00FF	Send 0 of codes	00000
LEAD	16660	byte	LEAD, 00FF	Send 0 of codes	00000
LEAD	16661	byte	LEAD, 00FF	Send 0 of codes	00000
LEAD	16662	byte	LEAD, 00FF	Send 0 of codes	00000
LEAD	16663	byte	LEAD, 00FF	Send 0 of codes	00000
LEAD	16664	byte	LEAD, 00FF	Send 0 of codes	00000
LEAD	16665	byte	LEAD, 00FF	Send 0 of codes	00000
LEAD	16666	byte	LEAD, 00FF	Send 0 of codes	00000
LEAD	16667	byte	LEAD, 00FF	Send 0 of codes	00000
LEAD	16668	byte	LEAD, 00FF	Send 0 of codes	00000
LEAD	16669	byte	LEAD, 00FF	Send 0 of codes	00000
LEAD	16670	byte	LEAD, 00FF	Send 0 of codes	00000
LEAD	16671	byte	LEAD, 00FF	Send 0 of codes	00000
LEAD	16672	byte	LEAD, 00FF	Send 0 of codes	00000
LEAD	16673	byte	LEAD, 00FF	Send 0 of codes	00000
LEAD	16674	byte	LEAD, 00FF	Send 0 of codes	00000
LEAD	16675	byte	LEAD, 00FF	Send 0 of codes	00000
LEAD	16676	byte	LEAD, 00FF	Send 0 of codes	00000
LEAD	16677	byte	LEAD, 00FF	Send 0 of codes	00000
LEAD	16678	byte	LEAD, 00FF	Send 0 of codes	00000
LEAD	16679	byte	LEAD, 00FF	Send 0 of codes	00000
LEAD	16680	byte	LEAD, 00FF	Send 0 of codes	00000
LEAD	16681	byte	LEAD, 00FF	Send 0 of codes	00000
LEAD	16682	byte	LEAD, 00FF	Send 0 of codes	00000
LEAD	16683	byte	LEAD, 00FF	Send 0 of codes	00000
LEAD	16684	byte	LEAD, 00FF	Send 0 of codes	00000
LEAD	16685	byte	LEAD, 00FF	Send 0 of codes	00000
LEAD	16686	byte	LEAD, 00FF	Send 0 of codes	00000
LEAD	16687	byte	LEAD, 00FF	Send 0 of codes	00000
LEAD	16688	byte	LEAD, 00FF	Send 0 of codes	00000
LEAD	16689	byte	LEAD, 00FF	Send 0 of codes	00000
LEAD	16690	byte	LEAD, 00FF	Send 0 of codes	00000
LEAD	16691	byte	LEAD, 00FF	Send 0 of codes	00000
LEAD	16692	byte	LEAD, 00FF	Send 0 of codes	00000
LEAD	16693	byte	LEAD, 00FF	Send 0 of codes	00000
LEAD	16694	byte	LEAD, 00FF	Send 0 of codes	00000
LEAD	16695	byte	LEAD, 00FF	Send 0 of codes	00000
LEAD	16696	byte	LEAD, 00FF	Send 0 of codes	00000
LEAD	16697	byte	LEAD, 00FF	Send 0 of codes	00000
LEAD	16698	byte	LEAD, 00FF	Send 0 of codes	00000
LEAD	16699	byte	LEAD, 00FF	Send 0 of codes	00000
LEAD	16700	byte	LEAD, 00FF	Send 0 of codes	00000
LEAD	16701	byte	LEAD, 00FF	Send 0 of codes	00000
LEAD	16702	byte	LEAD, 00FF	Send 0 of codes	00000
LEAD	16703	byte	LEAD, 00FF	Send 0 of codes	00000
LEAD	16704	byte	LEAD, 00FF	Send 0 of codes	00000
LEAD	16705	byte	LEAD, 00FF	Send 0 of codes	00000
LEAD	16706	byte	LEAD, 00FF	Send 0 of codes	00000
LEAD	16707	byte	LEAD, 00FF	Send 0 of codes	00000
LEAD	16708	byte	LEAD, 00FF	Send 0 of codes	00000
LEAD	16709	byte	LEAD, 00FF	Send 0 of codes	00000
LEAD	16710	byte	LEAD, 00FF	Send 0 of codes	00000
LEAD	16711	byte	LEAD, 00FF	Send 0 of codes	00000
LEAD	16712	byte	LEAD, 00FF	Send 0 of codes	00000
LEAD	16713	byte	LEAD, 00FF	Send 0 of codes	00000
LEAD	16714	byte	LEAD, 00FF	Send 0 of codes	00000
LEAD	16715	byte	LEAD, 00FF	Send 0 of codes	00000
LEAD	16716	byte	LEAD, 00FF	Send 0 of codes	00000
LEAD	16717	byte	LEAD, 00FF	Send 0 of codes	00000
LEAD	16718	byte	LEAD, 00FF	Send 0 of codes	00000
LEAD	16719	byte	LEAD, 00FF	Send 0 of codes	00000
LEAD	16720	byte	LEAD, 00FF	Send 0 of codes	00000
LEAD					